Additive models

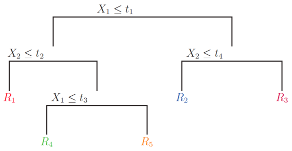
Manel Martínez-Ramón

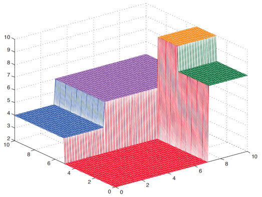
ECE, UNM

December 4, 2019

Trees

A Classification and Regression Tree (CART) is a recursive binary representation of the data





Root nodes contain a single feature, either X_1 or X_2. Leaf nodes contain partitions of an $\mathbb{R}^2$ space.

Trees

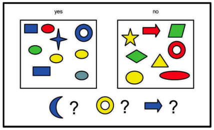
If $\mathbf{v}_m : \{X_m \leq t_m\}$, then the response of the tree can be coded as a linear combination of basis functions $\phi(\cdot)$

$$f(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = \sum_{m=1}^{M} w_m \mathbb{I}(\mathbf{x} \in R_m) = \sum_{m=1}^{M} w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

where R_m is a region and w_m is the mean response in this region.

Trees

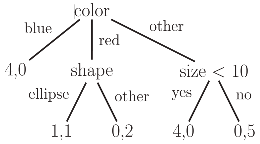
Classification trees.



D features (attributes)

N cases

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0



Root nodes contain a single feature. Leaf nodes contain values $[n_0, n_1]$ of instances of each class matching the feature value.

Tree growing algorithm

```
1  function fitTree(node, $\mathcal{D}$, depth);
2  node.prediction = mean($y_i : i \in \mathcal{D}$) // or class label distribution;
3  ($j^*$, $t^*$,$\mathcal{D}_L$,$\mathcal{D}_R$) = split($\mathcal{D}$);
4  **if** not *worthSplitting*(depth, cost, $\mathcal{D}_L$, $\mathcal{D}_R$) **then**
5      return node
6  **else**
7      node.test = $\lambda \mathbf{x}.x_{j^*} < t^*$ // anonymous function;
8      node.left = fitTree(node, $\mathcal{D}_L$, depth+1);
9      node.right = fitTree(node, $\mathcal{D}_R$, depth+1);
10     return node;
```

$\mathcal{D}$ refers to the training data set.

Split and *worthSplitting* functions

The split function chooses the best feature x_{ij} and the best value t according to a given cost function:

$$(j^*, t^*) = \arg \min_{j\in\{1\cdots D\}} \min_{t\in\mathcal{T}_j} \{\mathrm{cost}\left(\{\mathbf{x}_i, y_i; x_{ij} \le t\}\right) + \mathrm{cost}\left(\{\mathbf{x}_i, y_i; x_{ij} > t\}\right)\}$$

The *worthSplitting* function checks if a node is worth splitting using some heuristic:

- is the reduction in cost too small?
- has the tree exceeded the maximum desired depth?
- is the distribution of the response in either $\mathcal{D}_L$ or $\mathcal{D}_R$ sufficiently homogeneous (e.g., all labels are the same, so the distribution is pure)?
- is the number of examples in either $\mathcal{D}_L$ or $\mathcal{D}_R$ too small?

Regression cost

We define the cost as follows:

$$\text{cost}(\mathcal{D}) = \sum_{i \in \mathcal{D}} (y_i - \bar{y})^2$$

where $\bar{y}$ is the mean of the response variable in the specified set of data. Alternatively, we can fit a linear regression model for each leaf, using as inputs the features that were chosen on the path from the root, and then measure the residual error.

Classification cost

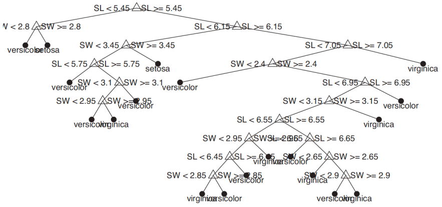
First, a Multinoulli distribution of the current split must be estimated for all classes:

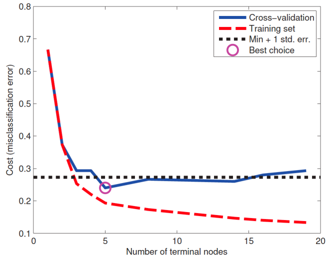
$$\hat{\pi}_c = p(y_i = c)$$

Then a probabilistic measure has to be computed:

- Probability of error: $1 - \arg\max_c \hat{\pi}_c$
- Entropy: $\mathbb{H}(\hat{\boldsymbol{\pi}}) = -\sum_{c=1}^{C} \hat{\pi}_c \log \hat{\pi}_c$
- Gini Index: $\sum_{c=1}^{C} \hat{\pi}_c (1 - \hat{\pi}_c)$

Example





Example of classification among three different classes of flowers (Iris database).

Advantages and trade offs

Advantages:

- Easy to interpret;
- Handle mixed discrete and continuous inputs;
- Insensitive to monotone transformations of the inputs;
- Automatic variable selection,
- Relatively robust to outliers,
- Scale well to large data sets, and
- they can be modified to handle missing inputs

Disadvantages:

- Not very accurate compared to other kinds of model.
- Trees are unstable: small changes to the input data can have large effects on the structure of the tree.

Random forests

In order to reduce the variance of trees, the bagging technique averages many different estimates:

$$f(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\mathbf{x})$$

where samples to train different trees are chosen randimly with repetition (Breiman, 1996).
Random forests use trees with different randomly chosen features and different random subsets (Breiman 2001)

Generalized additive models

A generalized additive model has the form

$$f(\mathbf{x}) = \alpha + \sum_{i=1}^{D} f_i(x_i)$$

where $f(x_i) = \beta_i \phi(x_i)$. The model can be applied to a cost function with a regularizer term

$$\sum_{i=1}^{N} e_i^2 + \sum_{j=1}^{D} \lambda_j \int \beta_j^2 \phi^2(t) dt$$

Generalized additive models

Interactions between variables can be added:

$$f(\mathbf{x}) = \alpha + \sum_{i=1}^{D} f_i(x_i) + \sum_{i,j} f_i(x_i, x_j) + \sum_{i,j,k} f_i(x_i, x_j, x_k) + \cdots$$

A particular case of this is the Volterra expansion of Module 5. In general, any type of spline functions can be used in this model.

Adaboost

Consider the additive model

$$f(\mathbf{x}) = \omega_0 + \sum_{m=1}^{m} \omega_m \phi_m(\mathbf{x})$$

and the exponential cost function

$$\exp(-y_i f(\mathbf{x}_i)), \quad y_i \in \{-1, 1\}$$

The minimization problem at step m is

$$L_m(\phi) = \sum_{i=1}^{N} \exp\left[-y_i \left(f_{m-1}(\mathbf{x}_i) + \beta_m \phi_m(\mathbf{x}_i)\right)\right]$$

This can be rewritten as

$$\begin{aligned} L_m(\phi) &= \sum_{i=1}^{N} \exp\left[-y_i\left(f_{m-1}(\mathbf{x}_i) + \beta_m \phi_m(\mathbf{x}_i)\right)\right] \\ &= \sum_{i=1}^{N} \omega_{i,m} \exp\left[-y_i \beta_m \phi_m(\mathbf{x}_i)\right] \end{aligned}$$

where

$$\begin{aligned} \omega_{i,m} =& \exp\left(-y_i\left(f_{m-1}(\mathbf{x}_i)\right)\right) \\ =& \omega_{i,m-1} \exp\left(-y_i \beta_m \left(\phi_m(\mathbf{x}_i)\right)\right) \end{aligned}$$

is a weight applied to sample i at iteration m.

The problem consists of minimizing

$$L_m(\phi) = \sum_{i=1}^{N} \omega_{i,m} \exp\left[-y_i \beta_m \phi_m(\mathbf{x}_i)\right]$$

Considering the convexity of the exponential:

$$\begin{aligned}
&\sum_{i=1}^{N} \omega_{i,m} \exp\left[-y_i \beta_m \phi_m(\mathbf{x}_i)\right] \\
&\leq \sum_{i=1}^{N} \frac{1 - y_i \phi_m(\mathbf{x}_i)}{2} \omega_{i,m} \exp(\beta_m) + \sum_{i=1}^{N} \frac{1 + y_i \phi_m(\mathbf{x}_i)}{2} \omega_{i,m} \exp(-\beta_m) \\
&= e_m \exp(\beta_m) + (1 - e_m) \exp(-\beta_m)
\end{aligned}$$

Nulling the derivative with respect to β_m gives its optimal value

$$\beta_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$

where $e_m = \sum_{i=1}^{N} \omega_{i,m} \mathbb{I}(y_i \neq \phi_m(\mathbf{x}_i))$

Adaboost Algorithm

1. Training data $\{\mathbf{x}_1; y_1 \cdots \mathbf{x}_N; y_N\}$
2. Set $\omega_{i,0} = \frac{1}{M}$
3. **for** m in $1 \cdots M$
    1. Train weak learner ϕ_m with distribution $\omega_{i,m}$
    2. Compute the weighted error $e_m = \sum_{i=1}^{N} \omega_{i,m} \mathbb{I}(y_i \neq \phi_m(\mathbf{x}_i))$
    3. Compute $\beta_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$
    4. Update weights $\omega_{i,m} = \omega_{i,m-1} \exp\left(-y_i \beta_m \left(\phi_m(\mathbf{x}_i)\right)\right)$
    5. Normalize $\omega_{i,m}$ so that $\sum_i \omega_{i,m} = 1$
4. Final hypothesis:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \beta_m \phi_m(\mathbf{x})\right)$$

Weak learners

A simple way to train a learner ϕ_m is to adjust a linear classifier using the distribution $\omega_{i,m}$. Thus, the learner is

$$\phi_m(\mathbf{x}_i) = sign(\mathbf{w}_m^\top \mathbf{x}_i)$$

where it is assumed that a bias is included in the weight vector. The optimization can be written as

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \omega_{i,m}(y_i - \mathbf{w}_m^\top \mathbf{x}_i)^2 = \min_{w} \|\mathbf{\Omega}_m^{\frac{1}{2}}\mathbf{y} - \mathbf{\Omega}_m^{\frac{1}{2}}\mathbf{X}^\top \mathbf{w}_m\|^2$$
$$= \min_{\mathbf{w}} \left(\mathbf{w}_m^\top \mathbf{X}\mathbf{\Omega}_m \mathbf{X}\mathbf{w}_m - \mathbf{w}_m^\top \mathbf{X}\mathbf{\Omega}_m \mathbf{y}\right)$$

where $\mathbf{\Omega}_m = \mathrm{diag}[\omega_{1,m} \cdots \omega_{N,m}]$ and whose solution is

$$\mathbf{w}_m = \left(\mathbf{X}\mathbf{\Omega}_m \mathbf{X}^\top\right)^{-1} \mathbf{X}\mathbf{\Omega}_m \mathbf{y}$$

Weak learners

- Nevertheless, a better solution is to simply choose base learners ϕ_m with VC dimension $h = 2$, this is, expressed in a subspace of one dimension.
- They can be seen as decision trees of one single node.

In order to train a tree, we must look for the optimal feature in $\mathbf{x}$ and its optimal threshold. A way to do it is

```
1  for $j$ in $1 \cdots D$
    1  sort all instances of feature $x_{i,j}$
    2  use them as threshold to split data $\mathbf{x}_i$ and compute the split error
    3  Choose the threshold that produces the minimum error.
2  Choose the dimension $j$ and threshold $t_j$ that produce the minimum error.
```

Outcomes of this lesson

- This lesson contains a summary of ensemble learning.
- The simplest adaptive basis function structure is the classification and regression tree (CART). In spite of its advantages, its performance is poor compared to other structures.
- Ensembles of random trees constructed with subsets of features are called randomm forests. They are known to have excellent performance at expenses of a considerable computational burden.
- An alternative ensemble learning methodology is boosting.
- The Adaboost algorithm combines weak learners. The learners are trained with the most difficult samples, and then they are weighted as a function of its quality.